

# Apple IIGS
# #90: 65816 Tips and Pitfalls

Revised by: Matt "Matt" Deatherage March 1991
Written by: Dave "Dave" Lyons September 1990

This Technical Note presents short 65816 assembly language examples illustrating pitfalls and clever techniques.

**Changes since November 1990:** Added more explanations about the JSL table and corrected a comment.

---

## Dispatching Through an Address Table

The 65816 has a `JSR ($aaaa,X)` instruction for calling a selected subroutine from a table of addresses, but it has no `JSL ($aaaa,X)` instruction. If you need to dispatch to one of several routines that are not all in the same bank, you need an approach like the following. The idea is to perform a `JSL` to a routine which does a long jump by pushing a three-byte "`RTL` address" on the stack and then doing an `RTL`.

```
            jsl LngJmp          ;go jump to the routine
            ...

  LngJmp    asl a               ;take routine number in A and
            asl a               ; multiply it by 4
            tax                 ;put table index into X
            lda table+1,x       ;get "middle" word of address
            pha                 ; and push it
            lda table,x         ;get low word and
            dec a               ; decrement it by one
            phb                 ;push a single throw-away byte
            sta 1,s             ;store over low two of the 3 bytes
            rtl                 ;transfer control to the routine
table       dc.l routine1       ;table of 4-byte subroutine addresses
            dc.l routine2
            dc.l routine3
            ...
```

This code is correct because `RTL` pulls three bytes off the stack and increments the two low bytes **without** incrementing the high byte.

**Note:** This approach to a table-based JSL is more flexible than `JML ($XXXX)` because it does not require any fixed-location storage or bank zero space, other than the stack.

On the other hand, the following code is not correct. The approach here is to make a table of addresses *minus* one.

```
        asl a             W
        asl a             R  ;multiply index by 4
        tax               O  ; and put it in X
        lda table+1,x     N  ;get the "middle" word
        pha               G  ; and push it
        lda table,x       !  ;get the low word
        phb               W  ;push a single throw-away byte
        sta 1,s           R  ;store over low two bytes
        rtl               O  ;transfer control to the routine
table   dc.l routine1-1   N  ;table of 4-byte addresses minus one
        dc.l routine2-1   G
        dc.l routine3-1   !
        ...
```

This second sample code fragment fails if any of the routines in the table comes at the first byte of a bank. For example, if `routine1` is at $060000, the address pushed is $05FFFF, and `RTL` transfers control to $050000, not $060000.

## Dereferencing Handles Without Direct Page Space

When your code gets called with the `D` register undefined, you must **not** use direct page addressing without setting `D` to a known good value. Preserving and restoring locations on the caller's direct page is **not** reliable, because `D` could be pointing at bytes below the stack pointer (which can be destroyed by interrupts) or even at the $C0xx soft switches (that would make your direct page accesses accidentally fiddle with hardware).

A common way to get temporary direct page space is to point `D` at part of your stack. This following code dereferences a handle stored in the `A` and `X` registers (if the handle is $E01234 and refers to a block of memory at $056789, then on entry `A`=$00E0 and `X`=$1234, and on exit `A`=$0005 and `X`=$6789).

```
        phd                 ;save caller's direct-page register
        pha                 ;push high word of handle
        phx                 ;push low word of handle
        tsc                 ;get stack pointer in A
        tcd                 ;and put it in D
        lda [1]             ;get low word of master pointer (no ",Y"!)
        tax                 ; and put it in X
        ldy #$0002          ;offset to high word of master pointer
        lda [1],y           ;get high word
        ply                 ;remove low word of handle
        ply                 ; and high word
        pld                 ;restore the caller's direct-page register
```

Direct page addressing isn't the only way to address through pointers. Here's the same routine as before, but using the Data Bank register (B) instead of fiddling with D. (Note that handles do **not** have to be in bank $E0 or $E1, although they usually are.)

```
        phb                 ;save caller's data bank register
        pha                 ;push high word of handle on stack
        plb                 ;sets B to the bank byte of the pointer
        lda |$0002,x        ;load the high word of the master pointer
        pha                 ; and save it on the stack
        lda |$0000,x        ;load the low word of the master pointer
        tax                 ;and return it in X
        pla                 ;restore the high word in A
        plb                 ;pull the handle's high word high byte off the stack
        plb                 ;restore the caller's data bank register
```

## Emulation Mode Has 65816 Features

You don't have to switch into Native mode just to do an eight-bit operation with long addressing. Most 65816-specific instructions and addressing modes work in emulation mode in approximately the same way they work in eight-bit native mode. See the "Further Reference" for details.

### Further Reference

- *Apple IIGS Hardware Reference*
- *Programming the 65816, Including the 6502, 65C02 and 65802* (Eyes and Lichty, 1986, Brady)